Abstract. This paper investigates a problem of sharing and reuse of innovative practices among teachers, rather than technical platforms. The paper starts from the emerging concept of learning designs (LDs) that are increasingly used to express teaching/learning scenarios that could be based on any pedagogical model (e.g. problem-based learning, role-playing etc). To address the identified research problem, the paper proposes to extend the existing model of LD lifecycle to capture various knowledge management processes necessary for sharing and reuse of innovative practices. To illustrate the proposed approach, the paper uses an example of problem-based learning. Finally the paper uses design research to set foundations for a new type of knowledge-management, educational technology that could be used to support knowledge sharing among teachers.

Keywords. Knowledge sharing, innovative teaching, learning designs, flexible processes

1. Introduction

Over the last decade, the field of computer-supported education has grown so much that e-learning is now being recognized as an emerging industry sector [6]. An ever increasing number of different educational systems and platforms has created the need for standardisation of content, tools and approaches. However, a large number of applications still concentrate on the educational content (teaching and learning resources). Only recently, researchers and industry practitioners are starting to recognise that the content-oriented view is quite limited. Consequently, there is an increased interest in the problem of sharing and reuse of learning experiences [14]. One of the most prominent research directions in this area focuses on the emerging theory of Learning Designs (LDs). Here, LDs are used to describe learning experiences in the form of highly specialised collaborative processes. In essence, a LD specifies “under which conditions, which activities have to be performed by learners and teachers to enable learners to attain the desired learning objectives” [8]. Typically, LDs combine technology-supported as well as human tasks and are expressed as high-level conceptual models. The well known examples of LDs include models of problem-based learning and various role-playing scenarios. In this paper, we distinguish between a conceptual model of learning design (LD) and formal learning design specification (IMS-LD) as defined by the IMS-LD standard [7].

The LD theory starts from the proven pedagogical models, rather than available technology. Its main objective is to describe the best practices how to achieve the intended set of learning objectives, in the given domain, in the most effective way, with or without technology. The ongoing process of design, implementation and continuous improvement of LDs is described by the term LD lifecycle.

This emerging area of educational research offers many open challenges related to both pedagogical aspects as well as possible IT support. Examples include: development of authoring environments and other tools for learning designs, identification of learning design patterns, user-friendly graphical notation of learning models, ontologies for specific pedagogical models, the problem of runtime and design adaptations, runtime collaborative service integration etc. [9].

However, it is important to observe that all these challenges deal with a lifecycle of a single LD, including mainly their modeling and execution/activation phases.

This paper goes one step further. It introduces a new LD research problem related to knowledge sharing and reuse of LDs among teachers. This is motivated by the real need to support sharing and
reuse of innovative teaching practices, especially in the emerging disciplines, such as business intelligence or business process management, where teaching practices are still evolving.

The proposed research problem is, in fact, a very challenging knowledge management (KM) problem, due to the number of reasons. LDs include tacit (experiential) knowledge-in-action, refined through years of teaching experiences and reflection in action. Design knowledge is always situational, as different methods are suitable for different teaching and learning situations [10]. Furthermore, LDs are not currently described in a way that is easy to communicate and share across disciplinary boundaries. For example, the existing formal models of IMS-LD are designed to be executed by computers and reused by different software platforms. As such, they are not suitable to be shared by teachers. Also, the existing tools, used to implement and support IMS-LD, are yet to support knowledge sharing activities.

The main objectives of this paper are twofold: - to further advance the theory of learning designs by proposing an extended model of the LD lifecycle that incorporates knowledge management processes related to sharing and reusing of innovative practices among teachers - to set foundations for a new type of KM technology designed to support teachers in their knowledge sharing processes.

The proposed approach is illustrated by an example of LD related to problem-based learning.

2. Related work

The last decade has seen many different types of web-based eLearning applications. The most popular category includes learning management systems (LMS) (e.g. Blackboard). Typically, LMS provide tools for authoring and presentation; assessment and feedback; student management as well as administrative and collaborative tools.

Despite their wide popularity, these systems are usually tightly coupled and vendor specific, with each vendor promoting their own solutions, making it very difficult for the users to use the components from other vendors (systems) [4]. In terms of process support, the latest enterprise version of LMS incorporates workflow functionality to enable integration of various aspects of the study process from the administrative point of view.

Another popular category of web-based educational environments includes Learning Content Management Systems (LCMS). The main objective of LCMS is to help create, reuse, locate and deliver online learning content, typically in the form of small, self-contained and self-describing units (or “chunks”) of instructional materials called learning objects. LCMS can help instructional designers to break the course content into learning objects and then deliver them in the “just-in-time and just-enough learning” mode [15]. The latest versions of popular LMS also use learning objects (for content representation and storage) and to some degree enable the same functionality as LCMS. It is expected that this two categories of systems will eventually merge.

The growing need to share and reuse content has resulted in the emergence of a number of important standards. For example, the LOM: Learning Object Metadata Standard [11] enables sharing of learning objects among content repositories. Although there are attempts to encourage direct sharing of learning objects among teachers (end-users) to facilitate knowledge sharing and reuse, they are primarily designed to be shared by software platforms.

Another important standard is certainly SCORM (Sharable Content Object Reference Model) [2]. Again, this standard is aimed at LCMS and in particular, sequencing of content.

Even though the theory and practice of LDs are relatively new, there are already notable developments of new standards as well as learning systems that either directly implement learning designs or follow the underlying principles of the LD theory. For example, the IMS Learning Design Specification [3] is a new standard, proposed to make formal modeling (digital representation) of learning designs consistent and consequently reusable by different software packages. Examples of various emerging systems include Coppercore, Edubox, Eduplone, LAMS, Lobster and Reload Software (see [3] and [10]).

These systems are designed to support the LD lifecycle with the main emphasis on design (modeling) and implementation (run-time) support. However, these systems still do not offer any knowledge management support to enable teachers to effectively share and reuse their learning designs. The following sections will describe this problem in more details.
3. An overview of learning designs

This section offers a brief overview of the LD theory, originally introduced by Koper and Olivier [9] and later extended by Koper and Tattersall [10]. Even though the central ideas behind LD theory and practice are not new, LDs are now more relevant than ever before, because of the wide use of content-oriented educational systems (or “electronic page turners”). These ideas can be briefly summarized as follows [3]:

- people learn better when they are actively involved in the learning process
- learning activities may be structured (e.g. sequenced) in a learning workflow
- it would be useful to be able to record learning designs (innovative practices) for future sharing and reuse.

While the increasing number of researchers are starting to use this concept of “learning workflow”, the previous applications of workflow-based systems in education (see for example [5], [12] and [15]), confirm that workflow’s way of sequencing learning activities is very inflexible. While this may be suitable in some situations, teaching/learning processes are, by nature, highly creative and should be flexible. Thus, in any teaching/learning situation, it is necessary to allow a teacher to modify the intended model or even start from an incomplete model, all while teaching is in progress. Consequently, the term “learning workflows” should be interpreted as flexible and descriptive rather than prescriptive sequences of learning tasks.

To make it easier to understand LDs, Koper and Olivier [6] use a metaphor of a theatrical play in order to illustrate the main ideas behind the learning/teaching process. The model of a play is a script that can be “instantiated” (staged) many different times by different actors in different environments and for different audiences. The actual performance (an instance of the process) is always unique. A play consists of one or more act(s) and an act is implemented by one or more concurrent roles playing different parts. The acts in a play can follow a sequence or a more complex structure that includes concurrent acts. Activities can be also assembled into more activity-structures that can include more complex coordination. During each act, roles use various resources to achieve the intended objectives.

Even though the concept of LD uses a “script”, it is meant to be less prescriptive and more flexible than the actual “theatrical script”. This means that all roles can create and implement new activities during the actual teaching/learning process. This will ensure that learning processes are truly flexible and driven by teachers.

As already pointed out, each LDs goes through a lifecycle that starts from the intended learning objectives and finishes with a reflection phase. Table 1 illustrates the key phases of a LD lifecycle and currently available support, as provided by the IMS-LD standard [7].

**Table 1. LD lifecycle [7]**

<table>
<thead>
<tr>
<th>Key phases</th>
<th>IMS-LD Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Learning Objectives</td>
<td>Specify Learning Objectives</td>
</tr>
<tr>
<td>Develop narrative description of learning and teaching scenario</td>
<td>Not defined within current scope</td>
</tr>
<tr>
<td>Create learning activity workflow from Narrative description</td>
<td>Create a Method using Play, Acts and Role-Parts</td>
</tr>
<tr>
<td>Assign resources, tools and people to activities</td>
<td>Specify Roles, Resources Environment and Services</td>
</tr>
<tr>
<td>Running (real-time)</td>
<td>Use a Learning Design aware player</td>
</tr>
<tr>
<td>Learner support and on-the-fly adaptation</td>
<td>Not Defined</td>
</tr>
<tr>
<td>Learner’s reflection (including sharing outputs for peer reflection)</td>
<td>Not Defined</td>
</tr>
</tbody>
</table>

4. Extending the LD lifecycle

The main objective of this section is to propose an extended model of LD lifecycle designed to incorporate various knowledge management processes that enable sharing and reuse of innovative practices.
The extended lifecycle also starts from the intended learning objectives and a conceptual model of LD used to describe a particular learning scenario. The next step is to analyse the LD in order to identify possible patterns of collaboration, communication and coordination tasks. Each pattern is then analysed to identify if it is possible to support it by the existing collaboration and communication tools (e.g. chat tools, forums). Alternatively, some patterns could lead to development of self-contained executable components designed to support learning tasks.

All identified components are then stored in a shared repository that is made available to teachers. The main idea is to enable teachers to design LDs by selecting and combining individual, executable components. These models of the assembled processes as well as the individual components are then made available to other teachers.

Analysis of a given LD and possible identification of new patterns and components are IT-related knowledge intensive tasks and should be done by knowledge engineer. On the other hand, composition, sharing and reuse of individual components should be driven by the intended LD should be left to the domain expert i.e. teacher.

A high-level model of the extended LD lifecycle is depicted by Fig 1. The labels numbered as 1-4, are used to represent the sequence of the LD design and reuse. It is clear that, to facilitate sharing of innovative practices both knowledge engineer and the teacher need engage in all four types of KM processes including (i) creation (construction), (ii) storage/retrieval (iii) transfer and (iv) application of knowledge. These KM processes were originally identified in the field of organizational knowledge [1] but are equally applicable here. The next section illustrates the proposed extended LD lifecycle by an example.

5. Sharing problem-based LDs

This section focuses on a LD based on the pedagogical model of problem-based learning. Suppose that students are given a problem solving assignment.

After creating and analyzing the formal model of this teaching/learning process, expressed as a LD, (outside of the scope of this paper due to limited space), knowledge engineer may observe various patterns of interactions (communication,
collaboration and coordination). That will lead to development of a number of executable components, such as:

Problem registration – This component enables teachers to post an assignment specification on the web. Students will be automatically notified that the assignment is ready via e-mails. This will also generate a temporal constraint (submission deadline) that is used for verification purposes during model composition as well as process instantiation.

Registration of groups – When this component is used, students will receive an invitation to register their groups by the given date. If configured in that way, this component may, for example, automatically form groups of all students who fail to register by the deadline.

Problem-solving component – This component is used to help participants generate and analyse different alternatives. It may include a number of subcomponents both generic and subject specific such as, brainstorming, mind-mapping etc.

Selection of a solution – This component may include support for voting, multiple-criteria decision making etc.

Assignment submission component – This could be designed to work similarly to conference paper submission system.

Peer-review component – This component could be implemented in a variety of ways to support different roles and coordinate tasks accordingly. For example, the same item can be marked only by the lecturer or by two different groups of students or even by an external marker.

It is important to observe that components are not simple task-oriented software tools activated during process execution to execute a single task and return a result. They encapsulate process-related concepts (such as pre-conditions, post-conditions, temporal constraints) that enable their assembly into a process. Furthermore, one component may incorporate a number of tasks, (such as the assignment submission component) and can use the alternative tools (resources) that can be chosen when process is assembled or executed.

Even with these few components teachers can create several different versions of the problemsolving process. For example, teacher A may assemble all the above components into a sequential process, giving her students a choice between various problem-solving subcomponents. At the same time teacher B may decide to use only “problem registration” and “assignment submission” components and supplement them with traditional face-to-face problem solving activities. Teacher C may request development of a new type of problem-solving component. Teacher D may decide to reuse the process created by teacher A and replace the electronic brainstorming tool with the electronic debate tool.

Furthermore, when creating learning designs some teachers may decide to assemble a complete model during design phase and then use it without any changes. Others may decide to use incomplete models (select only some components) and then complete them during runtime depending on students’ progress. They can even allow students to select their preferred components to support their own tasks.

6. Implementation Issues

The initial prototype of this technology called the web-based handbook of LDs was designed and implemented independently, while the development of the current version of the LD theory and IMS-LD standard were still in progress. The prototype included a repository implemented as a proprietary system based on the flexible component-based workflow technology. The repository was used to store a small number of components to support problem-based learning LD, as described here. For more details on technical solutions see [13].

The main objective of this prototype was to test the theoretical concepts and the proposed framework. The subsequent evaluation confirmed the fact that design and development of new technology, although very complex is still less complex than the creative challenge of identification of patterns that could be implemented as executable components. Having in mind that some of the identified components already exist as ready-made communication tools or recently, in the form of web-services, technology development is likely to become even less challenging. Rather than development of new components, the focus should be on possible reuse. This means that the main value-add, from the IT perspective, comes from possible integration of different components. This is also creates another IT challenge, as composition of different components should be done by end-users who are domain experts – in this case, teachers.

Most importantly, this paper envisages that the extended LD lifecycle, proposed here, can be
used as a methodological framework for future development of KM systems that is independent of any existing technology solutions and tools. This further reinforces the fundamental idea behind LD to start from pedagogical models and then determine possible support.

Finally, even if possible, not every aspect of a LD could be (and should be) supported by technology. Technology should not restrict flexibility of LD and teaching practices should not be controlled or driven by technology. Because, the main challenge of design of innovative practices to achieve the intended learning objectives will always remain more art than science.

7. Conclusions and future work

Learning designs describe “under which conditions, which activities have to be performed by learners and teachers to enable learners to attain the desired learning objectives” [8]. This is a relatively new area of educational research, and consequently, there are many open research challenges as described by [5]. This paper introduces a new LD research problem related to knowledge sharing and reuse of teaching/learning processes among teachers.

Current and future work focuses on design (science) research in this area. Challenges include further progress with new types of LDs, identification of learning patterns and identification and verification of executable components.

8. References


